Acionamento de comandos do carro via controle de voz

4º ponto de controle

Matheus Jericó Palhares
Universidade de Brasília - UnB
Faculdade Gama - FGA
Brasília, Brasil.
matheusjerico1994@hotmail.com

Lucas Raposo Souza Carvalho
Universidade de Brasília - UnB
Faculdade Gama - FGA
Brasília, Brasil.
lucas.raposo1995@gmail.com

Resumo: Dispositivo controlado por voz capaz de acionar algum comando, tendo como objetivo atuar no carro.

Palavras chaves: RaspBerry Pi, controle de voz, controle e automação de carro.

I. JUSTIFICATIVA

Com o aumento populacional, aumentou-se a quantidade de carros principalmente em grandes centros urbanos, causando grandes congestionamentos, estresse durante o caminho devido à quantidade de carros na pista, entre outros. Nessa linha de raciocínio, nota-se que é necessário cada vez mais prestar no trânsito enquanto dirige e muitas vezes não é possível realizar tal tarefa com excelência, visto que o motorista pode ser distraído devido à necessidade de ligar o farol do carro, aumentar o volume do som ou até escolher uma música infantil para acalmar a criança no banco de trás. Nesses pequenos momentos de distrações podem ocorrer acidentes como batidas e atropelamentos.

Um dispositivo que é controlado por voz (Jasper/Judy) e atua em vários sistemas do carro, resolveria grande parte dos problemas e ainda traz um conforto associado ao luxo extra ao motorista e aos passageiros do automóvel.

II. OBJETIVOS

Criar um dispositivo capaz de receber comando de voz de qualquer pessoa dentro do automóvel para realizar tarefas simples do veículo.

III. REQUISITOS

- Projeto deve ser feito em plataforma RaspBerry.
- Projeto deve ser capaz de ser implementado em qualquer veículo, mesmo após a montagem feita pela montadora.
- Projeto deve ser capaz de receber comando de qualquer tom de voz.
- O produto deve acionar após ser falada uma frase ou palavra específica evitando confusões de frases não relacionadas a instrução.
- O produto deve atender pelo menos os seguintes comandos: Ligar farol, ligar som, aumentar volume, ligar ar condicionado.

IV. BENEFÍCIOS

A implementação do projeto traz a facilidade de acionar ou ajustar comandos básicos do carro, sem que seja necessário retirar a atenção do trânsito, tarefa realizada apenas com o comando de voz. Tais tarefas que poderiam ocasionar acidentes graves caso não fossem feitas com a devida segurança.

V. IMPLEMENTAÇÃO

Foram utilizados os seguintes componentes:

- Raspberry pi.
- Modulo relé.
- Microfone USB (possui conversos A/D integrado).

Inicialmente, na implementação foi utilizada a biblioteca Jasper, que foi instalada na Raspberry Pi seguindo o tutorial do próprio site da Jasper. Pode-se configurar duas vertentes, Speech-To-Text (STT), que converte o comando de voz para texto, que no caso primeiramente utilizamos o Google API (STT), e podemos configurar o Text-To-Speech (TTS), que é exatamente o oposto do STT.

Após o microfone captar o “comando de voz”, ele faz a conversão do sinal analógico para o sinal digital. Dessa forma, a Jasper envia o “comando de voz” para a Google API via internet que é responsável por fazer a conversão do áudio para texto, quando concluído, é mandado de volta o texto convertido.

Após ter o áudio convertido para texto, o mesmo é comparado com as strings que foram adicionadas no arquivo "commands.conf", a Jasper faz a comparação do texto convertido com a string, conforme na figura abaixo:

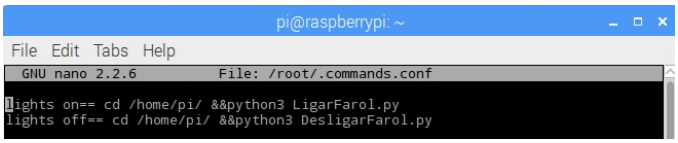

Figura 01: Arquivo commands.conf

Se o texto que foi convertido for "lights on", a Jasper vai abrir o arquivo em python chamado AcenderFarol.py, que fará a habilitação dos pinos de saída.

Já quando o texto convertido for "lights off", a Jasper vai abrir o arquivo em python chamado DesligarFarol.py, que será responsável por desabilitar os pinos.

Por enquanto, a Jasper só está atendendo falas em inglês porque o Google API está configurado para receber falas em inglês, mas será modificado para aceitar falas em português no próximo ponto de controle.

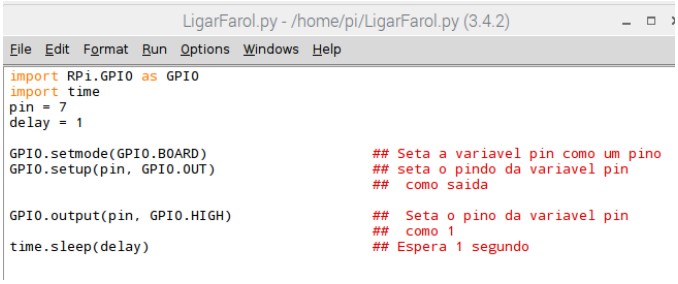

Figura 02: Arquivo AcenderFarol.py

Na figura acima observamos o código utilizado para habilitar o pino 7 de saída. Como o pino de saída da raspberry é 3.3 V, foi feita uma implementação em hardware para que a saída direcionada para o farol do carro seja de 12 V, pois é com 12 V que o farol do carro é acionado.

Para fazer a implementação em hardware, foi utilizado um transistor BC457 e um relé. O pino de saída em vez de estar conectado diretamente ao LED, está conectado ao transistor, que tem como função fazer chaveamento do relé.

Quando o pino 7 está em alto, seu valor é de 3.3V, como está conectado ao transistor no gate, o $V_{GS} > V_{TH}$, portanto o transistor vai conduzir corrente do dreno para source, fazendo com que o relé seja habilitado, fornecendo uma saída de 12 V.

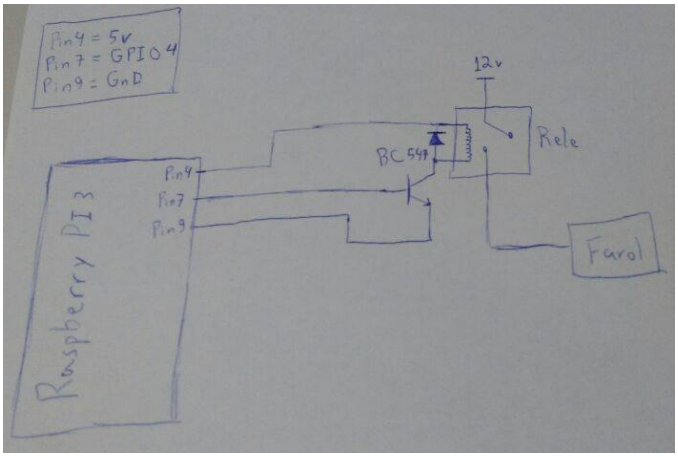

FIGURA 03: IMPLEMENTAÇÃO EM HARDWARE

Foi feita a troca do Controle de voz Jasper pelo Controle de voz Judy, para que as ferramentas de conversão texto/áudio fizessem a conversão sem utilizar a internet.

Dessa forma, foram configuradas as duas vertentes Speech-To-Text (STT) e Text-To-Speech (TTS), que não utilizam o artefato da internet para fazer a conversão. O STT é feito pelo Pocketsinx e o TTS é realizado pela Pico; toda conversão é feita offline.

Ainda no 2º ponto de controle, não tinha sido implementado a vertente TTS, portanto a RPi não se comunicava com o usuário. Para o 3º ponto de controle foi feita a implementação do TTS utilizando a Pico, dessa forma, a Rpi se comunica com o usuário.

O software criado é feito em python, importando a biblioteca Judy no início do programa.

```
                    lm='/home/pi/Documents/vocabulary/0112.lm',
                    dict='/home/pi/Documents/vocabulary/0112.dic')

vout = judy.VoiceOut(device='plughw:0,0',
                     resources='/home/pi/Documents/audio')

pygame.mixer.init()
pygame.mixer.music.load("Dont call me white - Nofx.mp3")

def handle(phrase):

    print 'Heard:', phrase

    if phrase == 'TURN LIGHTS ON':
        time.sleep(1)
        vout.say("LIGHTS ON")
        GPIO.output(7, GPIO.HIGH)

    if phrase == 'TURN LIGHTS OFF':
        time.sleep(1)
        vout.say("LIGHTS OFF")
        GPIO.output(7, GPIO.LOW)

    if phrase == 'PLAY MUSIC':
        time.sleep(1)
        vout.say("MUSIC ON")
        pygame.mixer.music.play()

    if phrase == 'STOP MUSIC':
        time.sleep(1)
        vout.say("MUSIC OFF")
        pygame.mixer.music.stop()
```

```
    if phrase == 'HIGHER MUSIC':
        time.sleep(1)
        volume = pygame.mixer.music.get_volume()
        print volume
        time.sleep(1)

        if volume == 1.0:
            time.sleep(1)
            vout.say("ALREADY IN THE HIGHEST")

        else:
            vout.say("CHANGE VOLUME")
            pygame.mixer.music.set_volume (volume+0.3)


    if phrase == 'LOWER MUSIC':
        time.sleep(1)
        volume = pygame.mixer.music.get_volume()
        print volume
        time.sleep(1)

        if volume == 0.0:
            time.sleep(1)
            vout.say("ALREADY IN THE LOWER")

        else:
            vout.say("CHANGE VOLUME")
            pygame.mixer.music.set_volume (volume-0.3)

judy.listen(vin, vout, handle)
```

Figura 04: Código em Python.

O programa faz a comparação do que foi falado no microfone com as strings definidas (“TURN LIGHTS ON”, PLAY MUSIC”, etc) se a comparação for verdadeira, a RPi utiliza o TTS para se comunicar com o usuário, emitindo a string desejada, logo após é realizado o acionamento do pino de saída.

Foi utilizada a biblioteca pygame, que emula uma central multimídia, dessa forma é possível realizar o acionamento da musica e o pause.

Para melhorar a acurácia do reconhecimento de voz, é criado um vocabulário de palavras onde o programa se atenta somente a essas palavras (arquivos .lm e .dic). As palavras são colocadas em um arquivo .txt que é enviado para uma universidade no EUA, onde é feita a conversão do arquivo .txt para os arquivos .lm e .dic, que são enviados de volta para RPi.

Figura 05: Arquivo .txt utilizado para confecção dos arquivos .lm e .dic.

O código em python que foi criado para implementação, utiliza os arquivos .lm e .dic para poder limitar o vocabulário da Rpi, e possui dois áudios de BIP como saída para indicar que está pronto para ouvir e que entendeu o que foi falado.

A implementação em hardware continua a mesma, foi alterado apenas a biblioteca de controle de voz, da Jasper para Judy, consequentemente os arquivos para configurar a mesma.

Referências Bibliográficas

[1] Jasper Project, configuração Jasper, disponível em: <www.jasperproject.github.io/documentation> Acesso em 30 de março de 2016.

[2] RaspBerry PI, controle de voz utilizando Siri, disponível em: <www.raspberrypi.org> Acesso em 30 de março de 2016

[3] Mitchell, M., Oldham, J. & Samuel, A., Advanced Linux Programming, Editora: Newriders, 2001.

[4] Source Forge, siri proxy, disponível em: <https://sourceforge.net/p/siriproxyrpi/wiki/Home/> Acesso em 30 de março de 2016.

[5] Judy Project, configuração Judy, disponivel em: <https://github.com/nickoala/judy> Acesso em 01 de junho de 2017.